Applied Soft Computing xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Comparisons of metaheuristic algorithms and fitness functions on software test data generation

Q1 Omur Sahin, Bahriye Akay*

Erciyes University, The Department of Computer Engineering, 38039 Melikgazi, Kayseri, Turkey

ARTICLE INFO

Article history:
Received 29 December 2015
Received in revised form 21 July 2016
Accepted 26 September 2016
Available online xxx

Keywords:
Software testing
Test data generation
Artificial Bee Colony
Particle Swarm Optimization
Differential Evolution
Firefly algorithm
Approximation level
Branch distance
Path-based coverage
Similarity-based coverage

ABSTRACT

Cost of testing activities is a major portion of the total cost of a software. In testing, generating test data is very important because the efficiency of testing is highly dependent on the data used in this phase. In search-based software testing, soft computing algorithms explore test data in order to maximize a coverage metric which can be considered as an optimization problem. In this paper, we employed some meta-heuristics (Artificial Bee Colony, Particle Swarm Optimization, Differential Evolution and Firefly Algorithms) and Random Search algorithm to solve this optimization problem. First, the dependency of the algorithms on the values of the control parameters was analyzed and suitable values for the control parameters were recommended. Algorithms were compared based on various fitness functions (path-based, dissimilarity-based and approximation level + branch distance) because the fitness function affects the behaviour of the algorithms in the search space. Results showed that meta-heuristics can be effectively used for hard problems and when the search space is large. Besides, approximation level + branch distance based fitness function is generally a good fitness function that guides the algorithms accurately.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Q2 Testing is an important phase in software production and software life cycle because testing cost is mostly accounted for 50% of the total development costs and releasing a reliable product is mostly depended on the testing activities. Of all the testing activities, test case generation involves major portion of the labour since it affects the efficiency of the whole software testing and then the software produced [1]. As a software becomes more complicated, software testing becomes more challenging [2]. In order to generate test data/case artifacts that make the testing more efficient and robust, a large number of tools have been proposed and used in the state-of-art for various programming languages, frameworks and platforms [1]. Anand et al. [1] categorized the most prominent techniques into five groups:

1. symbolic execution and program structural coverage testing
2. model-based test case generation
3. combinatorial testing
4. adaptive random testing
5. search-based testing

In search-based software testing (SBST), the main goal is to explore effective test data that maximizes a coverage metric of a software structure. Random testing [3] is a widely-used and low cost approach in which test inputs are randomly picked from valid ranges. However, its performance is poor when the inputs are subjected to hard constraints. In SBST, some meta-heuristic search techniques have been used for test case generation by formulating the problem as a combinatorial search problem. Harman and Jones [4] claims that search-based software testing is an emerging field and meta-heuristics are ideal to be applied software engineering by reformulating the classical software engineering problems. While reformulating the problem, a fitness function is defined to evaluate the quality of a solution in terms of a coverage metric. The meta-heuristics do not make assumptions on the problem characteristics and produce reasonable results for difficult problems that cannot be solved by analytical approaches because of the problem dimensionality, the problem surface that has discontinuities and noise. Search-based software test data generation methods are reviewed by Harman and Jones [4], McMinn [5], Afzal et al., Räih"a [6], McMinn [7], Harman et al. [8], and Harman et al. [9]. In all these reviews, it is stated that there remain plenty of fields related with

* Corresponding author.
E-mail addresses: omur@erciyes.edu.tr (O. Sahin), bahriye@erciyes.edu.tr (B. Akay).

http://dx.doi.org/10.1016/j.asoc.2016.09.045
1568-4946/© 2016 Elsevier B.V. All rights reserved.

UNCORRECTED PROOF

search-based software engineering and many interesting research challenges ahead.

There are some various meta-heuristic algorithms proposed based on different natural phenomena. Particle Swarm Optimization (PSO) algorithm [10], Differential Evolution (DE) [11], Artificial Bee Colony (ABC) [12], Firefly algorithm (FA) [13] are some examples of most popular meta-heuristic algorithms.

PSO algorithm [10], introduced by Eberhart and Kennedy in 1995, is a swarm-based meta-heuristic that models the social behaviour of bird flocking or fish schooling. DE algorithm [11] proposed by Storn and Kennedy for numerical optimization problems is a population based algorithm using evolutionary operators; crossover, mutation and selection. Artificial Bee Colony (ABC) algorithm [12] developed in 2005 by Karaboga mimics the foraging behaviour of honey bees and has been applied to many problems encountered in different research areas [14–16]. FA [13] presented by Yang based on the flashing and communication behaviour of fireflies with flashes.

In recent years, some studies on PSO, DE, ABC and FA for test data/case generation have been presented to the literature. Windisch et al. [17] combined PSO and branch coverage criteria to generate test data and conducted a comparison between PSO and Genetic Algorithm (GA) on a benchmark set. Their results showed that PSO outperformed GA in terms of effectiveness and efficiency. Tiwari et al. [18] applied a variant of PSO in the creation of new test data for modified code in regression testing. They reported that the proposed algorithm performed better in terms of code coverage capability compared to other existing PSO algorithms on five well known benchmark test functions. Zhu et al. [19] proposed an improved PSO algorithm which uses adaptive inertia weight. Results showed that proposed PSO algorithm had better performance compared to immune genetic and basic PSO algorithms. Dahiya et al. [20] proposed a hybrid pseudo dynamic testing based on PSO to generate test data for C programs using the all-path testing criterion. Experiments were carried out on the structural testing problems such as dynamic variables, input dependent array index, abstract function calls, infeasible paths and loop handling. The technique was claimed to be robust and to produce test inputs that are not redundant. Singla et al. [21] combined GA and PSO algorithm to generate automatic test data for data flow coverage with using dominance concept between two nodes on a number of programs having different size and complexity. It was stated that the performance of new technique is superior to both GA and PSO. Latiu et al. [22] performed a comparison between GA, PSO and Simulated Annealing algorithms that were integrated with the approximation level and branch distance metrics. The results indicated that evolutionary testing strategies are suitable to generate test data with a high coverage amount.

Landa Becerra et al. [23] built a test data generator which employed some DE variants and branch coverage metric for automated test data generation problem which can be formulated as a constrained optimization problem. DE algorithm was compared to the Breeder Genetic Algorithm and it was concluded that DE is a promising solution technique for this real-world problem. Jianfeng et al. [24] presented a DE-based combinatorial test data generator. In the approach, a selection and substitution based on the degree of unfinished interaction are employed to optimize the test case selected in further. They compared the proposed approach with Automatic Efficient Test Generator, Simulated Annealing, GA, Ant Colony Optimization, Cross-Entropy and PSO algorithms and the results showed the competitiveness of the proposed approach in test suite size and running time. Liang et al. [25] proposed a DE algorithm based on the one-test-at-a-time strategy for test case suite generation. In the experiments, the effect of different mutations and the influence of the control parameter values were analyzed. It was concluded that the approach was effective and improved the solution.

Mala et al. employed ABC algorithm for software test suite optimization and compared ABC and GA in [26], and ABC and ACO in [27]. Their fitness function was based on coverage-based test adequacy criteria. They reported that the proposed approach had less computation time, was more scalable and effective. In [28], they parallelized the approach in order to reduce time overhead and compared it to sequential ABC, GA and Random Testing. The results indicated that the proposed ABC-based approach converged within less number of test runs. Dahiya et al. [29] employed ABC with branch distance-based objective function for automatic test data generation in structural software tests and performed experiments on ten real world problems with large-range input variables. It was reported that the new technique was a reasonable alternative for test data generation but the performance was deteriorated when the inputs have large range and problem has many equality constraints. Lam et al. [30] presented a parallel ABC approach for the automated generation of feasible independent test path based on the priority of all edge coverage criteria to achieve the all test coverage with less number of test runs. It was stated that the proposed approach did not get stuck to local optima and path sequence comparison performed better than many fitness functions in literature. Malhotra and Khari [2] proposed different variants of ABC algorithm which utilizes mutation function of GA, in onlooker and scout bee phases in order to further improve the global search capability of the basic ABC algorithm. The proposed approach was evaluated on 10 C++ programs and it was shown that ABC algorithm with mutation produced better results in less time. Malhotra et al. [31] applied ABC, ant colony and GA algorithm based on path coverage metric and the experiments were validated on 9 C++ programs. It was concluded that ABC was the efficient due to the incorporation of parallelism and its neighbourhood production mechanism. Suri and Kaur [32] applied ABC algorithm as a regression test data generator to find the affected portions in a program and to achieve maximum path coverage. New test cases are generated until 100% coverage is achieved. Experiments were repeated for eight examples and the proposed approach was shown to be able to detect the paths that had been affected by changes and achieved 100% path coverage.

Srivatsava et al. [33] optimized test case paths using FA based on appropriate objective function and introducing guidance matrix in traversing the graph. Their objective function employed cyclomatic complexity and random function. Graph reduction and state-based transformation ensured the right code coverage for testing. It was shown that FA produced the optimal paths and could minimize the test efforts.

One purpose of this paper is to investigate the search abilities of PSO, DE, ABC, FA and Random Search algorithms on software test data generation benchmark problems including the triangle classifier, quadratic equation, even-odd, largest number, remainder, leap year, division of mark problems. PSO, DE, ABC and FA algorithms are simple and practical to implement compared to many other search heuristics. Especially ABC algorithm has little number of control parameters to be tuned. In most of the studies given above, there is no experiment on the control parameter sensitivity. Because the algorithm dependent parameters have critical impact on the local and global search abilities of the algorithms; they are directly related to the efficiency and efficacy. In this study, we conducted a comprehensive parametric analysis by setting different values for control parameters of PSO, DE, ABC and FA and suggested values for control parameters that yield generally good performance.

Besides, in the experiments, designing a fitness function is another key issue that should be decided [4]. Tailoring a good fitness function helps the algorithm to track the optima in the search space more accurately and quickly. In SBST tools based

UNCORRECTED PROOF

Please cite this article in press as: O. Sahin, B. Akay, Comparisons of metaheuristic algorithms and fitness functions on software test data generation, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.09.045

Download English Version:

https://daneshyari.com/en/article/4963629

Download Persian Version:

https://daneshyari.com/article/4963629

Daneshyari.com